REVERSIBLE DATA-EMBEDDING WITH A HIERARCHICAL STRUCTURE

Jun Tian and Raymond O. Wells, Jr.

School of Engineering and Science
International University Bremen
P. O. Box 750 561
28725 Bremen, Germany
juntian@ieee.org, wells@iu-bremen.de

ABSTRACT

Reversible data-embedding embeds data into a digital content (such as digital image, audio or video) such that an authorized party could decode the embedded data and completely restore the original content. In this paper, we present a hierarchically structured, multi-layer algorithm for digital images. Its basic component is an invertible sharpen filter called the difference expansion. The embedding capacity of this algorithm is substantially higher than results reported in previous reversible data-embedding algorithms. The visual quality of images after data-embedding is also among the best in the literature.

1. INTRODUCTION

Reversible data-embedding ([1, 2, 3, 4, 5, 6, 7, 8, 9]), embeds (hides) data (information) into a digital image, such that an authorized party could decode the embedded data and also restore the image to its original, pristine state. Traditional data-embedding (or data-hiding) is a lossy process with respect to the image: from the embedded image, which is the image after data-embedding, it is impossible to restore the original image. Reversible data-embedding is a lossless process: both the embedded data and original image could be decoded from the embedded image. Reversible data-embedding could be used as an information carrier and covert communication channel. It gives a self authentication scheme, by embedding the message authentication code of a digital image. It does not change the file syntax of a digital image; a legacy viewer can view the embedded image; the embedded data becomes an inherent part of the content, and is robust against (lossless) image format conversion.

2. THE DIFFERENCE EXPANSION TECHNIQUE

The difference expansion (DE) technique [6] reversibly embeds one bit data into two integers. It consists of the following three steps. For two integers x and y, first we define their integer average l and difference value h as

$$l := \left\lfloor \frac{x+y}{2} \right\rfloor, \; h := x - y, \tag{1}$$

where the symbol $\lfloor \cdot \rfloor$ is the floor function. The inverse transform of (1) is

$$x = l + \left\lfloor \frac{h+1}{2} \right\rfloor, \; y = l - \left\lfloor \frac{h}{2} \right\rfloor. \tag{2}$$

Second we embed the one bit data b, where $b = 0$ or 1, into the difference value h by a shift left and appending. The new difference value h' will be

$$h' = 2h + b.$$

Third and finally, we apply the inverse transform (2) to derive the embedded integers x' and y',

$$x' = l + \left\lfloor \frac{h'+1}{2} \right\rfloor, \; y' = l - \left\lfloor \frac{h'}{2} \right\rfloor.$$

When specified to 8 bits per pixel (bpp) grayscale images, x and y will be pixel values. As such pixel values are bounded in the range from 0 to 255, one needs to prevent overflow and/or underflow after data-embedding. By using (2), it can be easily checked that the necessary and sufficient condition to prevent overflow/underflow is

$$|h'| \leq \min(2(255 - l), 2l + 1),$$

that is,

$$|2h + b| \leq \min(2(255 - l), 2l + 1), \tag{3}$$

for both $b = 0$ and 1. If Condition (3) is satisfied, we say h is *expandable* (under l).

For a digital image, we put pixel values into pairs. Then we embed one bit into each of those expandable difference values. When the amount of the embedded data is less than the embedding capacity limit, one may select a portion of

0-7803-8554-3/04/$20.00 ©2004 IEEE.

expandable difference values for the DE. To assist the decoder to locate these expanded difference values, a location map (which is a mask with value "1" at expanded difference values) will also be embedded. From the filtering point of view, the DE is an invertible sharpen filter. For convenience, we call the embedded data the *payload*.

3. HIERARCHICALLY STRUCTURED MULTI-LAYER REVERSIBLE DATA-EMBEDDING

3.1. Encoding

The pseudo-code of an M-layer reversible data-embedding algorithm for an image I_0, where $M \geq 1$ is an integer, is:

1. Group pixel values of I_0 into pairs. Select a portion of expandable difference values. Create a location map l_1 whose value is "1" at a selected expandable h, and "0" at a not-selected expandable or not-expandable h. Losslessly compress l_1 and assume the compressed bit stream is $\mathcal{L}_1$.
2. Embed a payload $\mathcal{P}_1$ by embedding one bit of $\mathcal{P}_1$ into each selected expandable h by the DE. Assume the resulting image after data-embedding is I_1.
3. For $j = 2 : (M-1)$
   - Group pixel values of I_{j-1} into pairs. Select a portion of expandable difference values. Losslessly compress the location map l_j and assume the compressed bit stream is $\mathcal{L}_j$.
   - Embed $\mathcal{L}_{j-1}$ and a payload $\mathcal{P}_j$. Assume the resulting image after data-embedding is I_j.
4. Reversibly embed $\mathcal{L}_{M-1}$ and a payload $\mathcal{P}_M$ into I_{M-1}.

The last step, which is the M-th layer embedding, will be explained later.

The selection of expandable difference values determines the embedding capacity. To reach the embedding capacity limit, all expandable difference values are selected. When the amount of the embedded data is less than the embedding capacity limit, one selects a portion of expandable difference values for the DE. Two simple selection methods are proposed in [6]: one selects expandable difference values with small magnitudes; another selects expandable difference values with large hiding ability. No matter which criteria is used, the decoder can retrieve the location of expanded difference values from the location map. The location map is a one-bit bitmap. A value "1" indicates a selected expandable difference value. The location map can be losslessly compressed by a JBIG2 compression. An end of message symbol is at the end of the compressed bit stream $\mathcal{L}_j$.

Next we turn to the M-th layer. Unlike other layers, the compressed location map $\mathcal{L}_M$ of the M-th layer must be embedded into the M-th layer. As the embedded data by the DE is located at the least significant bits (LSBs), we can take the LSBs of all difference values into consideration. For a difference value that is not selected for the DE, we may modify its LSB as well. We define a singular set S as

$$S := \{(x, y) | x = 0 \text{ or } 255, \text{ and } y \text{ is an odd integer}\},$$

which is the collection of all pixel value pairs (x, y) that if we modify the LSB of its difference value, the new pair computed from (2) will have either an overflow or underflow. For convenience, we will say its difference value $h \in S$ if $(x, y) \in S$. We denote by R the complement set of S,

$$R := S^c = \{h | h \notin S\}.$$

Now we present the encoding algorithm at the M-th layer:

1. Group pixel values of I_{M-1} into pairs. Assume the collection of all difference values is $\{h_1, h_2, \cdots, h_n\}$. Denote by E the set of selected expandable difference values. Losslessly compress the location map l_M and assume the compressed bit stream is $\mathcal{L}_M$.
2. For each difference value $h \in (R \backslash E)$, LSB(h) will be collected into a bit stream $\mathcal{C}$.
3. Set $\mathcal{B} = \mathcal{L}_M \sqcup \mathcal{C} \sqcup \mathcal{L}_{M-1} \sqcup \mathcal{P}_M = b_1 b_2 \cdots b_m$, where $\sqcup$ is the concatenation operation, $b_i \in \{0, 1\}$.
4. Embed the bit stream $\mathcal{B}$ into difference values.
   (a) Set $i = 1$ and $k = 0$.
   (b) While $(i \leq m)$
      i. $k = k + 1$.
      ii. If $h_k \in E$
         - $h_k = 2h_k + b_i$.
         - $i = i + 1$.
      iii. Elseif $h_k \in (R \backslash E)$
         - $h_k = 2\left\lfloor \frac{h_k}{2} \right\rfloor + b_i$.
         - $i = i + 1$.
   (c) End
5. Apply the inverse transform (2) to obtain I_M.

After data-embedding, a difference value h in R will have its value staying outside the set S. An h in S will remain in S. The invariant feature of the set S (and equivalently R) is the key for the reversibility at the M-th layer.

3.2. Decoding

The decoding algorithm for the M-th layer is

1. Group pixel values of I_M into pairs, using the same grouping as in the encoding. For each difference value $h \in R$, LSB(h) will be collected into a bit stream $\mathcal{B}$.

Table 1. Embedded payload size vs. PSNR of embedded "Lena" images.

Payload Size (bits)	39566	63676	84066	101089	127133	170473	213212	260436	325062	507978	646221
Bit Rate (bpp)	0.1509	0.2429	0.3207	0.3856	0.4850	0.6503	0.8133	0.9935	1.2400	1.9378	2.4651
PSNR (dB)	44.20	42.86	41.55	40.06	37.73	36.42	34.93	32.57	29.67	23.57	16.09

2. Assume $\mathcal{B} = b_1 b_2 \cdots b_m$. Decode the location map l_M from $\mathcal{B}$. As $\mathcal{L}_M$ has an end of message symbol at its end, there is no ambiguity on the separation between $\mathcal{L}_M$ and the next bit stream segment, which would be $\mathcal{C}$. Assume $\mathcal{C}$ starts at the s-th bits in $\mathcal{B}$.

3. Restore original values of all difference values.

   (a) Set $i = s$.

   (b) For $k = 1 : n$

      i. If $h_k \in R$

         A. If the location map l_M has a value "1" at h_k

            - $h_k = \left\lfloor \frac{h_k}{2} \right\rfloor$.

         B. Else

            - $h_k = 2 \left\lfloor \frac{h_k}{2} \right\rfloor + b_i$.
            - $i = i + 1$.

   (c) End

4. Apply the inverse transform (2) to restore I_{M-1}.

5. From the remaining bits in $\mathcal{B}$, decode the location map l_{M-1} and payload $\mathcal{P}_M$.

If the location map value is "1", then the difference value h has been expanded, and an integer division by 2 will restore its original value. If the location map value is "0", and $h \in R$, its original value can be restored by restoring its original LSB.

The decoding of other layers is straightforward.

4. EXPERIMENTAL RESULTS AND CONCLUSION

We test the algorithm to various grayscale images. The embedded payload size and peak signal to noise ratio (PSNR) of embedded images for "Lena" are listed in Table 1. The payload size, which is $\sum_{j=1}^{M} |\mathcal{P}_j|$, does not include location maps, nor the bit stream $\mathcal{C}$. The original "Lena" image, and three embedded images are shown in Fig. 1. We are able to embed more than 2 bpp data into an 8 bpp grayscale image. Such an embedding capacity is substantially higher than results reported in existing algorithms. The visual quality of embedded images is also among the best in the literature.

In this paper, we have presented a low computational complexity, hierarchically structured reversible data-embedding algorithm for digital images. With the hierarchical structure, one may achieve scalable content access control by imposing different keys at different layers: a user with a high access level could decode all embedded data and restore the original, high quality image, yet a user with a low access level could only access the low quality image.

5. REFERENCES

[1] M. U. Celik, G. Sharma, A. M. Tekalp, and E. Saber, "Reversible data hiding," in *Proc. Int. Conf. Image Processing*, Sept. 2002, vol. II, pp. 157–160.

[2] J. Dittmann, M. Steinebach, and L. Ferri, "Watermarking protocols for authentication and ownership protection based on timestamps and holograms," in *Security and Watermarking of Multimedia Contents IV–Proc. SPIE*, E. J. Delp III and P. W. Wong, Eds., Jan. 2002, vol. 4675, pp. 240–251.

[3] J. Fridrich, M. Goljan, and R. Du, "Lossless data embedding–new paradigm in digital watermarking," *EURASIP J. Appl. Signal Processing*, vol. 2002, no. 2, pp. 185–196, Feb. 2002.

[4] C. W. Honsinger, P. W. Jones, M. Rabbani, and J. C. Stoffel, "Lossless recovery of an original image containing embedded data," *U.S. Patent, 6278791*, 2001.

[5] T. Kalker and F. M. J. Willems, "Capacity bounds and constructions for reversible data-hiding," in *Security and Watermarking of Multimedia Contents V–Proc. SPIE*, E. J. Delp III and P. W. Wong, Eds., Jan. 2003, vol. 5020, pp. 604–611.

[6] J. Tian, "Reversible data embedding using a difference expansion," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 8, pp. 890–896, Aug. 2003.

[7] A. van Leest, M. van der Veen, and F. Bruekers, "Reversible image watermarking," in *Proc. Int. Conf. Image Processing*, Sept. 2003, vol. II, pp. 731–734.

[8] C. De Vleeschouwer, J. F. Delaigle, and B. Macq, "Circular interpretation of bijective transformations in lossless watermarking for media asset management," *IEEE Trans. Multimedia*, vol. 5, no. 1, pp. 97–105, Mar. 2003.

[9] G. Xuan, J. Zhu, J. Chen, Y. Q. Shi, Z. Ni, and W. Su, "Distortionless data hiding based on integer wavelet transform," *IEE Electronics Letters*, vol. 38, no. 25, pp. 1646–1648, Dec. 2002.

(a) Original, 512×512, 8 bit grayscale "Lena".

(b) Reversibly embedded with a 39566 bits payload.

(c) Reversibly embedded with a 170473 bits payload.

(d) Reversibly embedded with a 646221 bits payload.

Fig. 1. Original and Reversibly Embedded "Lena".